Table of Contents

Memory Usage on Pleiades

Memory Usage Overview

Running jobs on cluster systems such as Pleiades requires more attention to the memory usage of a job than on shared memory systems. Below are a few factors that limit the amount of memory available to your running job:

- The total physical memory of a Pleiades compute node varies from 8 GB to 24 GB. A small amount of the physical memory is used by the system kernel. Through PBS, a job can access up to about 7.6 GB of an 8-GB node (Harpertown) and about 22.5 GB of a 24-GB node (Nehalem-EP and Westmere-EP).

- The PBS prologue tries to clean up the memory used by the previous job that ran on the nodes of your current running job. If there is a delay in flushing the previous job's data from memory to disks (for example, due to Lustre issues), the actual amount of free memory available to your job will be less.

- I/O uses buffer cache that also occupies memory. If your job does a large amount of I/O, the amount of memory left for your running processes will be less.

If your job uses more than 1 node, beware that the memory usage reported in the PBS output file is not the total memory usage for your job: rather, it is the *memory used in the first node* of your job. To help you get a more accurate picture of the memory usage of your job, we provide a few in-house tools, listed below.

1. ***qtop.pl*** invokes *top* on the compute nodes of a job, and provides a snapshot of the amount of used and free memory of the whole node and the amount used by each running process. For more information, read the article Checking Memory Usage of a Batch Job Using qtop.pl.

2. ***qps*** invokes *ps* on the compute nodes of a job, and provides a snapshot of the %mem used by its running processes. For more information, read the article Checking Memory Usage of a Batch Job Using qps.

3. ***qsh.pl*** can be used to invoke the command *cat /proc/meminfo* on the compute nodes to provide a snapshot of the total and free memory in each node. For more information, read the article Checking Memory Usage of a Batch Job Using qsh.pl and "cat /proc/meminfo".

4. ***gm.x*** and ***gm_post.x*** provide the memory high water mark for each process of your job when the job finishes. For more information, read the article Checking Memory Usage of a Batch Job Using qm.x.

These tools are installed under the directory /u/scicon/tools/bin. It is a good idea to include this directory in your path by modifying your shell startup script so that you don't have to provide the complete path name when using these tools. For example:

```
set path = ( $path /u/scicon/tools/bin )
```

If your job runs out of memory and is killed by the kernel, this event was probably recorded in system log files. Instructions on how to check whether this is the case are provided in the article Checking if a Job was Killed by the OOM Killer.

If your job needs more memory, read the article How to Get More Memory for your Job for possible approaches.

Checking memory usage of a batch job using qps

User Jeff West provided us with a Perl script called *qps* (available under /u/scicon/tools/bin) that securely connects (via ssh) into each node of a running job and gets process status (*ps)* information on each node.

Syntax:

```
pfe1% qps jobid
```

Example:

```
pfe1% qps 26130

*** Job 26130, User abc, Procs 1
NODE     TIME    %MEM %CPU STAT TASK
r1i0n14 10:17:13  2.8 99.9 RL   ./a.out
r1i0n14 10:17:12  2.9 99.9 RL   ./a.out
r1i0n14 10:17:18  2.9 99.9 RL   ./a.out
r1i0n14 10:16:34  2.9 99.8 RL   ./a.out
r1i0n14 10:17:11  2.9 99.9 RL   ./a.out
r1i0n14 10:17:13  2.9 99.9 RL   ./a.out
r1i0n14 10:17:12  2.9 99.9 RL   ./a.out
r1i0n14 10:17:15  2.9 99.9 RL   ./a.out
```

Note: The % memory usage by a process reported by this script is the percentage of memory in *the whole node*. This script currently works only when users specify `ncpus=8` in the PBS resource request.

If you want to use *qps* to monitor the memory used by a job that requested a number of CPUs other than 8, then make a copy of the qps script and change that single occurrence of '8' on line 95 to the appropriate number of CPUs requested on each node.

Checking memory usage pf a batch job using qtop.pl

DRAFT

This article is being reviewed for completeness and technical accuracy.

A Perl script called *qtop.pl* (available under /u/scicon/tools/bin) was provided by Bob Hood of the NAS staff. This script ssh's into the nodes of a PBS job and performs the command *top*. The output of qtop.pl provides memory usage for the whole node and for each process.

Syntax:

```
pfe1% qtop.pl [-b] [-p n] [-P s] [-h n] [-H s] [-t s] [-N s] PBSjobid
      -b   : (for running in background or batch) don't run 'resize' command
      -p n : show at most n processes per host
      -P s : show only procs in s, a comma-separated list of ranges
               e.g. -P 1,8-9
      -h   : don't show the column header line
      -H s : show only header lines in s, comma-separated ranges
               e.g. -H 1-2,7
               e.g. -H 0 (don't show any lines)
      -t s : pass string s (must be one argument) to top command
      -n s : show output only from nodes in s, comma-separated ranges
               e.g. -n 0,2-3           (relative node #'s)
      -N s : show output only from nodes in s, a comma-separated list
               e.g. -N r1i1n14,r1i1n15 (absolute node #'s)
```

Example: to skip the header and list 8 procs per host

```
pfe1% qtop.pl -H 0 -p 8 996093
all nodes in job 996093:  r184i2n12
r184i2n12   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
          20027 zsmith    25   0 23.8g 148m 5320 R  101  0.6   5172:37 a.out
          20028 zsmith    25   0 23.8g 140m 5140 R  101  0.6   5173:35 a.out
          20029 zsmith    25   0 23.9g 286m 6640 R  101  1.2   5172:23 a.out
          20030 zsmith    25   0 23.9g 245m 5040 R  101  1.0   5171:18 a.out
          20031 zsmith    25   0 23.9g 265m 6040 R  101  1.1   5171:46 a.out
          20032 zsmith    25   0 23.9g 246m 5300 R  101  1.0   5171:00 a.out
          20033 zsmith    25   0 23.8g 158m 5476 R  101  0.7   5172:41 a.out
          20034 zsmith    25   0 23.8g 148m 5280 R  101  0.6   5173:02 a.out
```

Checking memory usage of a batch job using qsh.pl and "cat /proc/meminfo"

DRAFT

This article is being reviewed for completeness and technical accuracy.

A Perl script called *qsh.pl* (available under /u/scicon/tools/bin) was provided by NAS staff member Bob Hood. This script ssh's into all the nodes used by a PBS job and runs a command that you supply.

Syntax:

```
pfe1% qsh.pl pbs_jobid command
```

One good use of this script is to check the amount of free memory in the nodes of your PBS job.

Example:

```
pfe1% qsh.pl 30329 "cat /proc/meminfo"

running "cat /proc/meminfo" on:  r56i2n14 r56i2n15
r56i2n14 :
  MemTotal:      8079728 kB
  MemFree:        857936 kB
  Buffers:             0 kB
  Cached:        3775472 kB
...
r56i2n15 :
  MemTotal:      8079728 kB
  MemFree:       5840920 kB
  Buffers:             0 kB
  Cached:         784280 kB
...
```

Checking memory usage of a batch job using gm.x

DRAFT

This article is being reviewed for completeness and technical accuracy.

NAS staff member Henry Jin created a tool called *gm.x* (available under /u/scicon/tools/bin) that reports the memory usage at the end of a run from each process.

Add */u/scicon/tools/bin* to your $PATH so that you can invoke *gm.x* without the full path.

Use the -h option to find out what types of memory usage can be reported:

```
pfe1%gm.x -h
gm - version 1.0
usage: gm.x [-opts] a.out [args]
   -hwm     ; high water mark (VmHWM)
   -rss     ; resident memory size (VmRSS)
   -wrss    ; weighted memory size (WRSS)
   -v       ; verbose flag
Default is by environment variable GM_TYPE (def=WRSS)
```

Note that the -rss option reports the last snapshot of resident set size usage captured by the kernel. With the -wrss option, *gm.x* calls the system function *get_weighted_memory_size*. More information about this function can be found from the man page **man get_weighted_memory_size**.

gm.x can be used for either OpenMP or MPI applications (linked with either SGI's MPT, MVAPICH or Intel MPI libraries) and you do not have to recompile your application for it. A script called *gm_post.x* then takes the per process memory usage information and computes the total memory used and the average memory used per process.

To use *gm.x* for an MPI code, add *gm.x* after the mpiexec options. For example:

```
mpiexec -np 4 gm.x ./a.out
Memory usage for (r1i1n0,pid=9767): 1.458 MB (rank=0)
Memory usage for (r1i1n0,pid=9768): 1.413 MB (rank=1)
Memory usage for (r1i1n0,pid=9770): 1.413 MB (rank=3)
Memory usage for (r1i1n0,pid=9769): 1.417 MB (rank=2)


mpiexec -np 4 gm.x ./a.out | gm_post.x
Number of nodes     = 1
Number of processes = 4
Processes per node  = 4
Total memory        = 5.701 MB

Memory per node     = 5.701 MB
Minimum node memory = 5.701 MB
```

```
Maximum node memory = 5.701 MB

Memory per process  = 1.425 MB
Minimum proc memory = 1.413 MB
Maximum proc memory = 1.458 MB
```

If you use dplace to pin process, add *gm.x* after dplace:

```
mpiexec –np NN dplace –s1 gm.x ./a.out
```

Checking if a Job was Killed by the OOM Killer

If a PBS job runs out of memory and is killed by the Out-Of-Memory (OOM) killer of the kernel, this event is likely (though not always) recorded in system log files. You can confirm this event by checking some of the messages recorded in system log files, and then increase your memory request in order to get your job running.

Follow the steps below to check whether your job has been killed by the OOM killer:

1. Find out when your job ran, what rack numbers were used by your job, and if the job exited with the Exit_status=137 from the tracejob output of your job. For example:

   ```
   pfe[1-12]% ssh pbspl1
   pbspl1% tracejob -n 3 140001
   ```

   where "3" indicates that you want to trace your job (PBS JOBID=140001), which ran within the past 3 days.

2. From the rack numbers (such as r2, r3, ...), you then grep messages that were recorded in the messages file stored in the leader node of those racks for your executable. For example, to look at messages for rack r2:

   ```
   pfe[1-12]% grep abc.exe /net/r2lead/var/log/messages
   Apr 21 00:32:50 r2i2n7 kernel: abc.exe invoked oom-killer:
   gfp_mask=0x201d2, order=0, oomkilladj=-17
   ```

3. Often, the Out-Of-Memory message doesn't make it into the messages file, but will be recorded in a consoles file named by each individual node. For example, to look for abc.exe invoking the OOM killer on node r2i2n7:

   ```
   pfe%  grep abc.exe /net/r2lead/var/log/consoles/r2i2n7
   abc.exe invoked oom-killer: gfp_mask=0x201d2, order=0, oomkilladj=0
   ```

   Note that these messages do not have a timestamp associated with them, so you will need to use an editor to view the file and look for the hourly time markers bracketing when the job ran out of memory. An hourly time marker looks like this:

   ```
   [-- MARK -- Thu Apr 21 00:00:00 2011]
   ```

It's also possible that a system process (such as, pbs_mom or ntpd) is listed as invoking the OOM killer, but it is nevertheless direct evidence that the node had run out of memory.

If you want to monitor the memory use of your job while it is running, you can use the tools listed in the article Memory Usage Overview.

In addition, NAS provides a script called *pbs_oom_check*. This script does the steps above and parses the /var/log/messages on all the nodes associated with pbs_jobid, looking for an instance of OOM killer. The script is available under /u/scicon/tools/bin and works best when run on the host pbspl1.

How to get more memory for your job

DRAFT

This article is being reviewed for completeness and technical accuracy.

If your job was terminated because it needed more memory than what's available in the nodes that it ran on, consider the following:

- Among the Harpertown nodes, the 64 nodes in rack 32 have 16 GB per node instead of 8 GB per node. You can request running your job on rack 32 with the keyword **bigmem=true**. For example, change

  ```
  #PBS -lselect=1:ncpus=8
  ```

  to

  ```
  #PBS -lselect=1:ncpus=8:bigmem=true
  ```
- Run your job on Nehalem-EP or Westmere nodes instead of Harpertown nodes. For example, change

  ```
  #PBS -lselect=1:ncpus=8:model=har
  ```

  to

  ```
  #PBS -lselect=1:ncpus=8:model=neh
  ```

  or

  ```
  #PBS -lselect=1:ncpus=8:model=wes
  ```

- If all processes use about the same amount of memory and you can not fit 8 processes per node (for Harpertown or Nehalem-EP, or 12 processes per node for Westmere-EP), reduce the number of processes per node and request more nodes for your job. For example, change

  ```
  #PBS -lselect=3:ncpus=8:mpiprocs=8:model=neh
  ```

  to

  ```
  #PBS -lselect=6:ncpus=4:mpiprocs=4:model=neh
  ```

- For a typical MPI job where rank 0 does the I/O and uses a lot of buffer cache, assign rank 0 to 1 node by itself. For example, change

```
#PBS -lselect=1:ncpus=8:mpiprocs=8:model=neh
```

to

```
#PBS
-lselect=1:ncpus=1:mpiprocs=1:model=neh+1:ncpus=7:mpiprocs=7:model=neh

Due to formatting issue, the above may appear as 2 lines. It should
really be just 1 line.
```

- If you suspect that certain nodes that your job ran on had less total physical memory than normal, report it to NAS Help Desk. Those nodes can be offlined and taken care of by NAS staff. This prevents you and other users from using those nodes before they are fixed.

- For certain pre- or post-processing work that needs more than 22.5 GB of memory, run it on the bridge nodes (bridge[1,2]) interactively. Note that jobs running on the bridge nodes can not use more than 48 GB of memory. Also MPI applications that use SGI's MPT library can not run on the bridge nodes.

- For a multi-process or multi-thread job, if any of your processes/threads needs more than 22.5 GB, it won't run on Pleiades. Run it on a shared memory system such as Columbia.